Lab 5 Supplement: Implementing an ALU

Prof. Onur Mutlu

ETH Zürich
Spring 2022
5 April 2022
What Will We Learn?

- In lab 5, you will **Implement an Arithmetic Logic Unit (ALU) in Verilog and evaluate its speed and resource utilization.**

- Draw a **block level diagram** of the MIPS 32-bit ALU, based on the description in the textbook.

- Implement the ALU using Verilog.

- Synthesize the ALU and evaluate **speed and FPGA resource utilization.**
Part 1: Designing an ALU

- We will design an ALU that can perform a subset of the ALU operations of a full MIPS ALU.
  - 2 32-bit inputs
  - 4-bit AluOp signal to select the operation
  - 32-bit output
  - Output flag zero that sets to logic-1 if all the bits of the result are 0.

<table>
<thead>
<tr>
<th>AluOp (3:0)</th>
<th>Mnemonic</th>
<th>Result =</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>add</td>
<td>A + B</td>
<td>Addition</td>
</tr>
<tr>
<td>0010</td>
<td>sub</td>
<td>A - B</td>
<td>Subtraction</td>
</tr>
<tr>
<td>0100</td>
<td>and</td>
<td>A and B</td>
<td>Logical and</td>
</tr>
<tr>
<td>0101</td>
<td>or</td>
<td>A or B</td>
<td>Logical or</td>
</tr>
<tr>
<td>0110</td>
<td>xor</td>
<td>A xor B</td>
<td>Exclusive or</td>
</tr>
<tr>
<td>0111</td>
<td>nor</td>
<td>A nor B</td>
<td>Logical nor</td>
</tr>
<tr>
<td>1010</td>
<td>slt</td>
<td>(A - B)[31]</td>
<td>Set less than</td>
</tr>
<tr>
<td>Others</td>
<td>n.a.</td>
<td>Don’t care</td>
<td></td>
</tr>
</tbody>
</table>
Part 1: Designing an ALU- Block Diagram

First, you need to draw a block diagram of the ALU, like the one seen in Figure 5.15 of the H&H textbook.

You are free to choose if you want to draw the complete block diagram or split it up into modules and elaborate the modules, similar to the next slide.
Part 1: Designing an ALU - Block Diagram

- A possible division in ALU Logic and Arithmetic operations:
Part 1: Designing an ALU- Block Diagram

- A possible organization of ADD and SUB:
Part 1: Designing an ALU - Block Diagram

- A possible organization for **SLT**: 

![ALU Block Diagram](image-url)
Part 2: Implementation

- Replace each block with a Verilog description.

- **Synthesize and implement** your design.

- We do not transfer the design to FPGA in this lab
  - No Constraint file → Bitstream generation will fail.

- At this point, we cannot verify the correctness of our circuit manually.
  - You will calculate how long will the exhaustive search take.
  - You learn how to use testbench to test the correctness of this circuit in lab 6! 😊
Part 3: The performance of the circuit (I)

- In this lab, we will learn to check:
  - The **speed** (i.e., max frequency our circuit can run at)
  - The **area** (i.e., FPGA resource utilization).

- We will add a timing constraint to set the **maximum delay** that we would like our ALU to have.
Part 3: The performance of the circuit (II)

The information we will obtain:

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of LUTs</td>
<td></td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
<td></td>
</tr>
<tr>
<td>Which pin of the FPGA is the output ‘zero’ connected? (pin name)</td>
<td></td>
</tr>
<tr>
<td>Where does the longest path start from</td>
<td></td>
</tr>
<tr>
<td>Where does the longest path end</td>
<td></td>
</tr>
<tr>
<td>How long is the longest path</td>
<td></td>
</tr>
<tr>
<td>How much of the longest path is routing</td>
<td></td>
</tr>
<tr>
<td>How many levels of logic is in the longest path</td>
<td></td>
</tr>
</tbody>
</table>
In lab 5, you will **Implement an Arithmetic Logic Unit (ALU) in Verilog** and evaluate its **speed and resource utilization**.

- Draw a **block level diagram** of the MIPS 32-bit ALU, based on the description in the textbook.

- Implement the ALU using Verilog.

- Synthesize the ALU and evaluate **speed and FPGA resource utilization**.

- In the report, you will use your adder from Lab 2 in the ALU and compare the resource utilization.
Report Deadline

23:59, 22 April 2022